TANDBERG

Reference guide

Codian Remote Management API

Document version 3.0

September 2008

TANDBERG

See: performance

TANDBERG
Philip Pedersens vei 20
1366 Lysaker
Norway
Telephone: +47 67 125 125
Telefax: +47 67 125 234
Video: +47 67 117 777
E-mail: tandberg@tandberg.com
www.tandberg.com

Copyright © TANDBERG 2008

Contents

1 Introduction 1
1.1 HTTP and HTTPS 1
1.2 XML-RPC 1
2 Protocol overview 2
2.1 Authentication 2
2.2 Message flow 2
2.3 Unicode support 4
2.3.1 HTTP Headers 4
2.3.2 XML Header 4
2.4 Common message elements 4
2.4.1 Authentication 4
2.4.2 Participant identification parameters 5
2.4.3 Enumerate functions 5
2.4.4 Filtering 6
3 Messages supported by multiple product types 8
3.1 device.query 8
3.2 device.network.query 8
3.3 device.health.query 9
3.4 device.restartlog.query 9
3.5 gatekeeper.query 10
3.6 sip.query 10
3.7 addressBookEntry.enumerate 11
3.8 gateway.enumerate 12
4 Conference related methods 14
4.1 conference.create 14
4.2 conference.modify 16
4.3 conference.destroy 17
4.4 conference.end 17
4.5 conference.enumerate 17
4.6 conference.streaming.query 20
4.7 conference.streaming.modify 21
4.8 conference.paneplacement.query 21
4.9 conference.paneplacement.modify 22
4.10 participant.add 22
4.11 participant.remove 24
4.12 participant.modify 25
4.13 participant.connect 26
4.14 participant.disconnect 26
4.15 participant.move 26
4.16 participant.enumerate 27
4.17 participant.fecc 31
4.18 participant.message 31
4.19 participant.diagnostics 32
4.20 autoAttendant.enumerate 32
4.21 autoAttendant.destroy 33
5 IP VCR methods 34
5.1 recording.callout 34
5.2 recording.configure 34
5.3 recording.delete 35
5.4 recording.enumerate 35
5.5 recording.stop 36
5.6 folder.enumerate 36

Copyright © TANDBERG 2008

6 ISDN Gateway methods 38
6.1 Common structures 38
6.2 calls.active.enumerate 39
6.3 calls.completed.enumerate 39
6.4 isdn.ports.query 40
7 IP Gateway methods 42
7.1 corpdirURI.query 42
7.2 corpdirURI.configure 42
8 Deprecated messages 43
8.1 system.query 43
8.2 conference.query 43
8.3 conference.participant.modify 44
8.4 conference.participant.remove 44
8.5 conference.participant.add 44
8.6 participant.enumerate 44
9 Related information sources 47
9.1 system.xml 47
10 Required user privileges 48
11 Fault codes 49
12 Participant disconnect reasons 51
13 References52
Appendix A - Conference layouts 53
Appendix B - Linking conferences across MCUs 55
B.1 Example message 1 - creating conference "linked1" on "mcu1" 55
B.2 Example message 2 - creating conference "linked2" on "mcu2" 56
B.3 Example message 3 - calling into "linked2" from "linked1" 57
B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout 58
B.5 Message responses 59
Appendix C - Revision Numbers 60
C.1 Discovering record removal 60
C.2 Dead records 61
Appendix D - HTTP Keep-alives 62

Copyright © TANDBERG 2008

1 Introduction

This document contains the specification of the TANDBERG Codian Remote Management API, by which it is possible to control several Codian products. This is accomplished via messages sent using the XML-RPC protocol.

XML-RPC is a simple protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, whilst allowing complex data structures to be transmitted, processed and returned. XML-RPC has no platform or software dependence and was chosen over SOAP because of its simplicity.

The interface is stateless. Currently, there is no mechanism for the Codian device to call back the controlling application and therefore the controlling application must poll the device for status, as required. A future enhancement *may* provide a mechanism for signaling device status changes to the controlling application.

The latest version of the Remote Management API is version 2.5, the following table shows which version of Codian products support this version.

API Version	**MCU 4200 MCU 4500 Media blades**	**IP VCR 2200 Recording blade**	**Codian ISDN Gateway 3200, 3201, MSE 8320, MSE 8321**	**IP Gateway 3500 Series IP Gateway MSE 8350 blade**
2.4	2.2	2.2	1.3	
2.5	2.3, and later	2.3, and later	1.4	2.0

1.1 HTTP and HTTPS

Codian devices expect to receive HTTP communication over TCP/IP connections to port 80. The HTTP messages should be “POST”s to the URL “/RPC2”.

HTTPS (a secure, encrypted version of HTTP) is supported on the following products:

- Codian MCU products, software version 2.3 and later
- Codian IP VCR products, software version 2.3 and later
- Codian ISDN GW products, software version 1.4 and later
- Codian IP GW products, software version 2.0 and later

By default, HTTPS is provided on TCP port 443, although Codian devices can be configured to receive HTTP and HTTPS connections on non-standard TCP port numbers if required.

The Codian devices implements HTTP/1.1 as defined by RFC 2616 [2].

1.2 XML-RPC

For the background and details of XML-RPC, please refer to the specification [1].

In this implementation, all parameters and return values are part of a <struct> and are all explicitly named. For example, the “device.query” method returns the current time value as a structure member named ‘currentTime’ rather than as a single value of type <dateTime.iso8601>.

2 Protocol overview

2.1 Authentication

In order to manage the device, the controlling application must authenticate itself as a user with relevant privileges. Accordingly, each message contains a user name and password; see section 2.4.1 for details of the format. It is worth noting that authentication information is sent using plain text and should only be sent over a trusted network.

2.2 Message flow

An application can create and manage conferences by sending command messages to the device. For each command sent (provided the message is correctly formatted according to the XML-RPC spec), the device responds with a message indicating success or failure. The response message may also contain any data that was requested.

Command messages are sent in XML format. For example, the following message schedules a conference on an MCU to begin at 10:45 on 18 February 2005 and last for one hour:

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: 10.2.1.100
Content-Type: text/xml
Content-length: 713

<?xml version="1.0"?>
<methodCall>
<methodName>conference.create</methodName>
<params>
<param>
<value>
<struct>
<member>
<name>authenticationUser</name>
<value>
<string>api_test</string>
</value>
</member>
<member>
<name>authenticationPassword</name>
<value>
<string>123456</string>
</value>
</member>
<member>
<name>conferenceName</name>
<value>
<string>Meeting 1</string>
</value>
</member>
<member>
<name>startTime</name>
<value>
<dateTime.iso8601>20050218T10:45:00</dateTime.iso8601>
</value>
</member>
```

```
<member>
<name>durationSeconds</name>
<value>
<int>3600</int>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

If the command was successful, the MCU sends a success response. For example, in response to a successful conference.create message, the MCU returns:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 240

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value>
<struct>
<member>
<name>status</name>
<value>
<string>operation successful</string>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

If the command fails, the MCU sends a fault response. For example, in response to a "conference.create" message where the conference name is not unique, the MCU returns:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 411

<?xml version="1.0"?>
<methodResponse>
<fault>
  <value>
    <struct>
      <member>
        <name>faultCode</name>
        <value>
          <int>2</int>
        </value>
```

```
      </member>
      <member>
        <name>faultString</name>
        <value>
          <string>duplicate conference name</string>
        </value>
      </member>
    </struct>
  </value>
</fault>
</methodResponse>
```

The complete list of command messages, their required and optional parameters, and the expected responses are detailed in the sections below. The possible fault codes are listed in section 11. Appendix B - contains examples of some messages and their corresponding responses.

2.3 Unicode support

Parameters in this version of the API can be in ASCII text or unicode (UTF8). In order to distinguish between these encodings, any of several methods can be used. If no method is present, ASCII is assumed.

2.3.1 HTTP Headers

There are two different ways of specifying unicode in the HTTP headers; either using `"Accept-Encoding: utf-8"`, or modifying the Content-Type header to read `"Content-Type: text/xml; charset=utf-8"`.

2.3.2 XML Header

At the top of each XML file, the <?xml> tag is required. This API will accept an additional encoding parameter with value UTF-8 for this tag, i.e. `<?xml version="1.0" encoding="UTF-8"?>`.

2.4 Common message elements

2.4.1 Authentication

All messages must contain a user name and password as follows:

Parameter	Type	Comments
authenticationUser	String	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
authenticationPassword	String	The corresponding user's password. This parameter is ignored if the user has no password set - note that this differs from the web interface where a blank password must be blank.

2.4.2 Participant identification parameters

The following parameters appear in the majority of conference control messages, and identify a specific participant on which operations are to be performed. The use of "MCU" below refers to any device which acts as a videoconferencing server.

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>participantName</td><td>String</td><td>This is an "internal" name, and therefore is not necessarily related to any name configured on an endpoint. Within the scope of a particular conference or auto attendant, the combination of "participantType", "participantProtocol" and "participantName" is always unique</td></tr>
<tr><td>participantProtocol</td><td>String</td><td rowspan="2">Used in conjunction with "participantName" to uniquely identify a participant within a connection. Typically, these parameters should be treated as opaque values, but the current possibilities are:
- for "participantProtocol":
`h323` – an endpoint using the H.323 protocol
`vnc` – a VNC connection (e.g. remote desktop)
`sip` – an endpoint using the SIP protocol.
- for "participantType":
`ad_hoc` – this participant called into the MCU or was dialed out via the web interface and is not in the MCU's endpoint list
`by_address` – fully-specified participant added through the API
`by_name` – MCU-configured endpoint irrespective of whether the endpoint dialed in or the MCU dialed out
API-created participants in scheduled conferences (i.e. those originated by "participant.add") will be of type `by_address` (unless they're added explicitly as temporary `ad_hoc` participants.)</td></tr>
<tr><td>participantType</td><td>String</td></tr>
<tr><td>conferenceName</td><td>String</td><td>Unique conference name – the conference name space is shared between API-generated conferences and all other ad hoc and scheduled MCU conferences.</td></tr>
<tr><td>autoAttendantUniqueId</td><td>String</td><td>If the participant in question is connected to an auto attendant rather than a conference, this field contains a unique identifier for that auto attendant.</td></tr>
</table>

When modifying or querying parameters for a specific endpoint, **participantName**, **participantProtocol** and **participantType** parameters are supplied, along with either a **conferenceName** or an **autoAttendantUniqueId**. The only safe way to find these values is to use the fields returned from participant.enumerate.

2.4.3 Enumerate functions

Due to the potential for a very large number of responses, all enumerate functions return an **enumerateID** response. This contains a string value which should be passed to subsequent calls of the same enumerate function in order to retrieve the remainder of the values.

The use of this parameter is as follows:

1. The client computer sends an enumerate call with any necessary parameters (e.g. **operationScope**) and no **enumerateID** parameter.
2. The device returns with an array containing the requested data, and possibly a new **enumerateID**.

3. If there is an **enumerateID**, the client should call the enumerate method again, with any parameters that are required or desired, and an **enumerateID** parameter containing the ID returned by the device from the previous call. This should be repeated while the device continues to provide new **enumerateID** values in responses.
4. After all data is returned, the device will reply with all remaining results, but no **enumerateID**.

This method should only be called using **enumerateID** values as provided by the device.

2.4.4 Filtering

Enumerate functions contain an optional **enumerateFilter** parameter, which can be used to restrict the responses to the enumerate call. The valid expressions depend on the function to which they are applied, but the syntax is the same for all enumerate functions: the section in this document for each function provides a list of valid filters for that function.

To use the filters, the expression is evaluated, with any function or expression symbols evaluated for the given entity being enumerated (e.g. if enumerating conferences, the *active* expression will evaluate to true if the conference is active, and false otherwise). If the result of evaluating the filter is true, the entity is returned to the client. If the expression evaluates to false, the entity in question is not returned to the client and the next entity (if any) is considered. As an example, if the expression (*active* && *scheduled)* is used when enumerating conference, the returned conferences will be only those which are both active and scheduled.

Filters can consist of atomic expressions, joined together with operators, and brackets in the traditional manner. Whitespace is ignored. Functions are valid, and any parameters are in a comma separated list in brackets following the function name, for example *function(expression*$_1$*,expression2)*.

From a boolean perspective, the integer 0 is false, and all other numbers are true.

Integer values can be expressed using any string of valid digits, optionally prefixed by 0x for hexadecimal, 0t for decimal and 0z for binary. If no prefix is specified, decimal is assumed.

The following binary operators are valid, in order of priority (lowest priority first)

Operator	**Description**
||	Boolean or
&&	Boolean and
|	Bitwise or
^	Bitwise exclusive or
&	Bitwise and
==	Equality
!=	Inequality
<	Less than
<=	Less than or equal
>=	Greater than or equal
>	Greater than
<<	Bitwise left shift
>>	Bitwise right shift
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo

There are also the following unary operators, all of which bind tighter than any binary operator.

Operator	Description
-	Unary minus
+	Unary plus
!	Logical negation
~	Bitwise negation

An example filter would be !(*expression*$_1$ && *expression*$_2$)

3 Messages supported by multiple product types

The methods in this section are common to many devices, including video conferencing servers, IP VCR products, ISDN GW products, and IP GW products. Not all methods are supported by all product types, and not all fields will be present in responses from all products.

3.1 device.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system's current time (UTC).
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.
serial	String	The serial number of the device.
softwareVersion	String	The software version of the running software.
buildVersion	String	The build version of the running software.
model	String	The model of this device, e.g. "Codian MCU 4210".
apiVersion	String	The version number of the API implemented by this device.
activatedFeatures	array of structs	Currently, only contains a string "feature" with a short description of the feature.
totalVideoPorts	Integer	The total number of video ports on the device. Only present on MCU and VCR products.
totalAudioOnlyPorts	Integer	The total number of additional audio-only ports on the device. Only present on MCU and VCR products.
portReservationMode	String	**`enabled`** or **`disabled`**, depending on the Media Port Reservation configuration setting. Only present on MCU products.
maxVideoResolution	String	One of **`cif`** or **`4cif`**. Only present on MCU and VCR products.
isdnPorts	Integer	The number of ISDN ports. Only present on ISDN gateways.

3.2 device.network.query

This call takes no parameters. The response returns the following:

Parameter	Type	Comments
portA	struct (see below)	Contains the configuration and status for port A.
portB	struct (see below)	Contains the configuration and status for port B.

The format for the two structures above is:

Field	Type	Comments
enabled	Boolean	true if the port is enabled, otherwise false.
hostName (optional)	String	The host name of this port.
dhcp (optional)	Boolean	true if configured by DHCP, otherwise false.
ipAddress (optional)	String	a.b.c.d format.
subnetMask (optional)	String	a.b.c.d format.
defaultGateway (optional)	String	a.b.c.d format.

Field	Type	Comments
domainName (optional)	String	The domain name of this port.
nameServer (optional)	String	a.b.c.d format.
nameServerSecondary (optional)	String	a.b.c.d format.
linkStatus	Boolean	true if the link is up, false if the link is down.
speed	Integer	one of 10, 100 or 1000, in Mbps.
fullDuplex	Boolean	true if full duplex enabled, false if half.
macAddress	String	a 12 character string, no separators.
packetsSent	Integer	Stats from the web interface. It is worth noting that all these values are 32 bit signed integers, and thus may wrap.
packetsReceived	Integer	
multicastPacketsSent	Integer	
multicastPacketsReceived	Integer	
bytesSent	Integer	
bytesReceived	Integer	
queueDrops	Integer	
collisions	Integer	
transmitErrors	Integer	
receiveErrors	Integer	
bytesSent64	String	64 bit versions of the above stats, using a string rather than an integer.
bytesReceived64	String	

All fields above marked optional will be returned only if the interface has been enabled and has been configured.

3.3 device.health.query

Returns the current status of the MCU, such as health monitors and CPU load.

Response	Type	Comments
cpuLoad	Integer	The CPU load, as a percentage.
mediaLoad	Integer	Loads for the media processors (total, and split between audio and video) as percentage values. These are not present on ISDN gateways.
audioLoad	Integer	
videoLoad	Integer	
fanStatus	String	One of **`ok`**, **`outOfSpec`** and **`critical`**.
fanStatusWorst	String	
temperatureStatus	String	
temperatureStatusWorst	String	
rtcBatteryStatus	String	
rtcBatteryStatusWorst	String	
voltagesStatus	String	
voltagesStatusWorst	String	
operationalStatus	String	One of **`active`**, **`shuttingDown`** or **`shutDown`**.

3.4 device.restartlog.query

Used to return the restart log (also known as the system log on the web interface).

Response	Type	Comments
log	array	Contains the restart log in structures as described below.

The "log" array consists of structures which contain the following fields.

Field	Type	Comments
time	dateTime.iso8601	The time of the last reboot.
reason	String	The reason for the reboot (one of **`unknown`**, **`User requested shutdown`** or **`User requested upgrade`**).

3.5 gatekeeper.query

Retrieves the gatekeeper settings and current status for an MCU or IP VCR. Takes no parameters.

Response	Type	Comments
gatekeeperUsage	String	One of **`disabled`**, **`enabled`** or **`required`**.
The following parameters are all optional and will only be present if gatekeeperUsage is not **`disabled.`**		
address	String	The address of the gatekeeper.
dnsStatus	String	The status of the DNS resolution: one of **`inProgress`**, **`resolved`** or **`failed`**.
ip	String	If the dnsStatus is **`resolved`**, contains the IP address of the gatekeeper.
activeRegistrations	Integer	The number of active registration.
pendingRegistrations	Integer	The number of registrations in progress.
registrationPrefix	String	The registration prefix used by the device
h323ID	String	The h323 id used by the device.
mcuServicePrefix	String	The service prefix used by the device.
scheduledConferenceIDRegistration	String	The value enabled or disabled; corresponds to web interface "ID registration for scheduled conferences" option. This field is only present for MCU product types, and only for MCU software versions 2.2(1.3) onwards.
h323IDStatus	String	The current status of the ID/service prefix registration process. One of **`idle`**, **`registering`**, **`registered`**, **`deregistering`**, **`pendingReregistration`**, **`waitingRetry`**, **`noID`** or **`idTooLong`**.
mcuServicePrefixStatus	String	

3.6 sip.query

Retrieves information on SIP configuration for an MCU or IP VCR. Takes no parameters.

Response	Type	Comments
configuredRegistrar	String	The currently configured SIP registrar address. This corresponds to the “SIP registrar address” on the **Settings >SIP** web page, and will be an empty string value if there is no currently configured SIP registrar.
configuredProxy	String	The currently configured SIP proxy address. This corresponds to the “SIP proxy address” on the **Settings > SIP** web page, and will be an empty string value if there is no

Response	Type	Comments
		currently configured SIP proxy.
conferenceRegistration	String	This value is only present if the device being queried is an MCU. It will be **enabled** if the MCU is configured to register conferences' numeric IDs with the configured SIP registrar, and **disabled** if not. The **enabled** value corresponds to "SIP registration settings" being set to "Allow conference registration" on the **Settings > SIP** page.

3.7 addressBookEntry.enumerate

Enumerates configured endpoints on an MCU or IP VCR.

Parameter	Type	Comments
enumerateID (optional)	String	The value returned by the last enumeration call. If it is omitted, a new enumeration is started.

This method returns:

Response	Type	Comments
enumerateID (optional)	String	The value that should be used in the next call to get the next set of data. If this is omitted, no further data is available from the MCU.
addressBookEntries	array of structs	See below for details.

The array "addressBookEntries" contains structs with the following fields:

Field	Type	Comments
Name	String	The configuration's name.
address	String (< 32 chars)	The participant's E.164 directory number, hostname or IP address.
protocol	String	One of `h323`, `sip` or `vnc`.
gatewayName	String	Present for h323 endpoints which are configured to use a gateway. This name corresponds to the name of a gateway returned via gateway.enumerate.
gatewayAddress	String	Present for h323 endpoints which are configured to use a gateway. This is the address of the gateway this endpoint is configured to use.
useSIPRegistrar	Boolean	Whether this endpoint is configured to use a sip registrar when being called.
password	String	The password for vnc endpoints.
portNumber	Integer	The port number for vnc endpoints.
callInParams	struct	See below for details.
conferencingParameters	struct	See below for details.

The structure callInParams contains the following fields. This is used to match incoming participants to endpoint configurations. For a positive match a participant must match fields which have values. Blank fields are not considered in the comparison.

Field	Type	Comments
name	String	Endpoint name.
address	String	IP address.
e164	String	E.164 number.

The structure conferencingParameters contains the following fields:

Field	Type	Comments
useDefaultMotionSharpness	Boolean	If true, this endpoint will use box-wide default motion sharpness settings.
minFrameRateMotionSharpness	Integer	Only present if useDefaultMotionSharpness is false. Specifies the minimum frame rate for this endpoint.
useDefaultVideoTransmitResolutions	Boolean	If true, this endpoint will use box-wide default video transmit resolutions.
videoTransmitResolutions	String	One of: **`allowAll, 4to3Only, 4to3WidescreenOverride`** or **`16to9Only.`**
maxMediaTxBitRate	Integer	Max media transmit bit rate.
maxMediaRxBitRate	Integer	Max media receive bit rate.
defaultLayout	String	Refer to appendix A for a list of layouts.
layoutControlDefault	Boolean	If true, this endpoint will use box-wide layout control settings.
layoutControlEnabled	Boolean	Only present if layoutControlDefault is false. Indicates whether the participant will have control over their layout.
h239ContributionDefault	Boolean	If true, this endpoint will use box-wide h239 contribution settings.
h239ContributionEnabled	Boolean	Only present if h239ContributionDefault is false. Specifies whether the endpoint will be able contribute h239.
initialAudioMuted	Boolean	Whether this participant would initially have their audio muted.
initialVideoMuted	Boolean	Whether this participant would initially have their video muted.
autoDisconnect	Boolean	When a participant disconnects from a conference and only participants who have autoDisconnect set to true remain, all those participants are disconnected.
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.

3.8 gateway.enumerate

Enumerates configured H.323 gateways on an MCU or IP VCR.

Parameter	Type	Comments
enumerateID (optional)	String	The value returned by the last enumeration call. If it is omitted, a new enumeration is started.

This method returns:

Response	Type	Comments
enumerateID (optional)	String	The value which should be used in the next call to get the next set of data. If this is omitted, no further data is available from the MCU.
gateways	array of structs	See below for details.

The array “gateways” contains structs with the following fields:

Field	Type	Comments
name	String	The name of the configured gateway.
address	String (< 32 chars)	The gateway’s E.164 directory number, hostname or IP address.
conferencingParameters	struct	see below

The structure conferencingParameters contains the following fields:

Field	Type	Comments
useDefaultMotionSharpness	Boolean	If true, this endpoint will use box-wide default motion sharpness settings.
minFrameRateMotionSharpness	Integer	Only present if useDefaultMotionSharpness is false. Specifies the minimum frame rate for this endpoint.
maxMediaTxBitRate	Integer	Max media transmit bit rate.
maxMediaRxBitRate	Integer	Max media receive bit rate.

4 Conference related methods

Methods in this section are only implemented on products, such as the Codian MCU 4200 series, which act as a video conferencing server. Throughout this section, the term MCU refers to any such device.

4.1 *conference.create*

Parameter	Type	Comments
conferenceName	String (<32 chars)	Name of the conference to be created. The conference name must be unique.
numericId (optional)	String (<32 chars)	Numeric identifier of the conference.
conferenceId **(deprecated)**	String (< 32 chars)	Deprecated alternative for "numericId".
registerWithGatekeeper (optional)	Boolean	Register the conference's "numericId" with the gatekeeper.
registerWithSIPRegistrar (optional)	Boolean	Register this conference with the SIP registrar.
startTime (optional)	dateTime.iso8601	If you do not specify a time, the conference starts immediately.
durationSeconds (optional)	Integer	The length of each repeating conference instance, in seconds. If this parameter is absent, or set to "0", the conference is permanent.
endTime (optional) **(deprecated)**	dateTime.iso8601	If you do not specify an end time, then the conference will be permanent (until it is explicitly deleted). **This parameter is deprecated and present for backward compatibility reasons only. Application code should use "durationSeconds" instead.**
pin (optional)	String (< 32 chars)	If present, this is the string of numeric digits that people need to enter to join the conference.
description(optional)	String (< 32 chars)	
multicastStreamingEnabled(optional)	Boolean	
unicastStreamingEnabled(optional)	Boolean	
h239Enabled(optional)	Boolean	
private	Boolean	Determines the visibility of this conference. This parameter corresponds to the "Visibility" setting on the web UI, which can have the value **`Public`** or **`Private`**.
maximumAudioPorts (optional)	Integer	These fields set the limit on the number of audio (audio only) and video (video + audio) ports for the conference. The
maximumVideoPorts (optional)	Integer	
reservedAudioPorts (optional)	Integer	

Parameter	Type	Comments
reservedVideoPorts (optional)	Integar	
repetition(optional)	String	One of: **none**, **daily**, **weekly**, **everyTwoWeeks** or **monthly**.
weekDay(optional)	String	Must be present if repetition is **monthly**. One of **monday**, **tuesday**, **wednesday**, **thursday**, **friday**, **saturday** or **sunday**. Note that if repetition is not **weekly** or **everyTwoWeeks**, the "weekDays" parameter should be used.
whichWeek(optional)	String	Must be present if repetition is **monthly**. One of: **first** (the first X of the month, where X is the day specified by weekday), **second**, **third**, **fourth**, or **last** (i.e. last X of the month).
weekDays(optional)	String	Must be present if repetition is **weekly** or **everyTwoWeeks**. A comma separated string of weekdays (i.e. any of **monday**, **tuesday**, **wednesday**, **thursday**, **friday**, **saturday** or **sunday**), e.g. **monday,wednesday,friday**.
terminationType (optional)	String	One of: **noTermination**, **afterNRepeats** or **endOnGivenDate**.
terminationDate (optional)	dateTime.iso8601	If terminationType is **endOnGivenDate**, this is the day that the conference repetition will end on.
numberOfRepeats (optional)	Integer	If terminationType is **afterNRepeats**, this is the number of repeats to end after.
customLayoutEnabled (optional)	Boolean	true if the layout is enabled, false otherwise.
newParticipantsCustomLayout (optional)	Boolean	true if new participants use the custom layout, false otherwise. Only valid if customLayoutEnabled is true.
customLayout (optional)	Integer	A layout index, as described in appendix A.

Parameter	Type	Comments
chairControl (optional)	String	The chair control setting for the conference. This can be **`none`**, **`floorControlOnly`** or **`chairAndFloorControl`** - these values correspond to the web interface "Floor and chair control" setting values of "Do not allow floor or chair control", "Allow floor control only" and "Allow floor and chair control" respectively. If not specified, the chair control setting for the new conference will be "Allow floor control only". This field is only present for MCU software versions 2.2(1.6) onwards.

Conferences created through the management API will appear in the list of conferences accessible via the web interface, and vice versa.

4.2 conference.modify

Parameter	Type	Comments
conferenceName	String (< 32 chars)	Name of the conference to modify.
newConferenceName (optional)	String (< 32 chars)	If present, the conference will be renamed to specified value.
oldConferenceName **(deprecated)**	String (<32 chars)	Deprecated conference renaming scheme – new code should use conferenceName and newConferenceName as above.
conferenceName **(deprecated)**	String (<32 chars)	
numericId	String (< 32 chars)	Optional fields as per "conference.create" described above. These fields can only be used for conferences which are not of type **`ad_hoc`**.
conferenceId **(deprecated)**	String	
pin	String	
registerWithGatekeeper	Boolean	
registerWithSIPRegistrar	Boolean	
startTime	dateTime.iso8601	
durationSeconds	Integer	
endTime **(deprecated)**	dateTime.iso8601	
Description	String	
multicastStreamingEnabled	Boolean	
unicastStreamingEnabled	Boolean	
h239Enabled	Boolean	
private	Boolean	
reservedVideoPorts	Integer	
reservedAudioPorts	Integer	
maximumVideoPorts	Integer	
maximumAudioPorts	Integer	
repetition	String	
weekDay	String	
whichWeek	String	

Parameter	Type	Comments
weekDays	String	
terminationType	String	
terminationDate	dateTime.iso8601	
numberOfRepeats	Integer	
h239Important	Boolean	Optional. Sets the h239 channel to be important.
locked	Boolean	Optional. Locks or unlocks the conference.
customLayoutEnabled	Boolean	Optional fields, as for the "conference.create" method above.
newParticipantsCustomLayout	Boolean	
customLayout	Integer	
chairControl	String	
enforceMaximumAudioPorts	Boolean	Assumed to be true if absent. These can be set to false in order to specify no limit on the number of audio/video ports.
enforceMaximumVideoPorts	Boolean	

Conferences created through the management API will appear in the list of conferences accessible via the web interface. Therefore, the API can be used to modify conferences scheduled via the web interface, and vice versa. Note that there is only a very limited amount of control available for **ad_hoc** conferences, however.

4.3 conference.destroy

Parameter	Type	Comments
conferenceName	String	Name of the conference to be destroyed.

A conference can be destroyed at any time; that is, before the conference has begun, during the conference or after the conference has ended. Destroyed conferences are removed entirely from the system; this includes all future repetitions of the conference.

4.4 conference.end

Parameter	Type	Comments
conferenceName	String	Name of the conference to be ended.

A conference remains in the list of conferences even after the conference has ended — until conference.destroy is called. In particular, this can be used to end an instance of a conference without deleting all future repetitions.

4.5 conference.enumerate

The conference enumerate function is used to return some or all conferences scheduled, running or completed on the MCU.

Parameter	Type	Comments
enumerateID (optional)	String	The value returned by the last enumeration call. If it is omitted, a new enumeration is started.
enumerateFilter	String	A filter expression.

Copyright © TANDBERG 2008

Valid expressions within the enumerate filter are as follows:

Expression	Type	Comments
active	Boolean	True if the conference is active.
completed	Boolean	True if the conference has finished.
scheduled	Boolean	True if the conference is a scheduled conference (regardless of if it has been completed or not).

This method returns:

Response	Type	Comments
enumerateID (optional)	String	The value which should be used in the next call to get the next set of data. If this is omitted, no further data is available from the MCU.
conferences	array of structs	See below for details.

The array "conferences" contains structs with the following fields:

Field	Type	Comments
conferenceName	String	
conferenceType	String	One of: `scheduled` or `ad_hoc`.
uniqueId	Integer	An id unique among all scheduled and ad hoc conferences, each instantiation of a scheduled conference will have the same uniqueId.
conferenceActive	Boolean	Indicates whether conference is currently active.
description	String	Extra user-specified information about the conference
pin	String	The security PIN.
guestPin	String	Security pin for a guest
numericId	String	
guestNumericId	String	
registerWithGatekeeper	Boolean	
registerWithSIPRegistrar	Boolean	
multicastStreamingEnabled	Boolean	
unicastStreamingEnabled	Boolean	
h239Enabled	Boolean	
h239Important	Boolean	Whether the H.239 channel is set to be important.
locked	Boolean	Whether the conference is locked or unlocked.
maximumAudioPorts	Integer	These fields set the limit on the number of audio (audio only) and video (video + audio) ports for the conference. The "reserved" values are for port reservation mode, whereas the "maximum" figures apply to non-reserved mode (and will be absent if no limits have been configured).
maximumVideoPorts	Integer	
reservedAudioPorts	Integer	
reservedVideoPorts	Integer	
customLayoutEnabled	Boolean	True if a custom layout has been enabled for this conference.
customLayout (optional)	Integer	The index (from appendix A) of the custom layout. This is only present if the custom

Field	Type	Comments
		layout is enabled.
newParticipantsCustomLayout	Boolean	True if new participants will use the conference custom layout.
private	Boolean	True if this conference is a private conference.
chairControl	String	The chair control setting for this conference. See the *chairControl* description in conference.create for an explanation of the values this parameter can take.
The following timing fields will be present for scheduled conferences only		
startTime	dateTime.iso8601	The time at which the conference started at or will start at.
durationSeconds	Integer	How long each repeating instance of the conference should last for. If absent, the conference is permanent.
repetition(optional)	String	One of **`none`**, **`daily`**, **`weekly`**, **`everyTwoWeeks`** or **`monthly.`**
weekDay(optional)	String	Present if repetition is **`monthly`**. One of **`monday`**, **`tuesday`**, **`wednesday`**, **`thursday`**, **`friday`**, **`saturday`** or **`sunday`**.
whichWeek (optional)	String	Present if repetition is **`monthly`**. One of: **`first`** (the first X of the month, where X is the day specified by weekday), **`second`**, **`third`**, **`fourth`**, or **`last`** (i.e. last X of the month)
weekDays (optional)	String	A comma separated string of weekdays (i.e. any of **`monday`**, **`tuesday`**, **`wednesday`**, **`thursday`**, **`friday`**, **`saturday`** or **`sunday`**), e.g. **`monday,wednesday,friday`**. This field is present when repetition is **`weekly`** or **`everyTwoWeeks`**.
terminationType(optional)	String	One of: **`noTermination`**, **`afterNRepeats`** or **`endOnGivenDate`**.
terminationDate(optional)	dateTime.iso8601	If terminationType is **`endOnGivenDay`**, this is the day that the conference repetition will end on.
The following timing values will be present for active conferences only		
activeStartTime	dateTime.iso8601	If the conference is currently active, these fields show the time span of the current activation. If the conference is permanent then "activeEndTime" will be absent.
activeEndTime	dateTime.iso8601	
activeConferenceId	string	A unique ID for the active instance of this conference; this conference will have this ID even if, for example, the conference is renamed while active, but each scheduled instance of this conference will have a different activeConferenceId.

4.6 conference.streaming.query

This returns some details on the current state of streaming viewers for a conference.

Parameter	Type	Comments
conferenceName	String	Name of the conference from which streaming information is required.

This will return a structure with the following fields:

Response	Type	Comments
unicastViewers	Integer	The number of unicast streaming viewers.
multicastViewers	Integer	The number of multicast streaming viewers.
audioStreams (optional)	Array	An array of stream structs (defined below). These are only present if there are any streams of either type currently in use.
videoStreams (optional)	Array	
audioRTCPReceiverReports	Integer	The number of RTCP receiver reports for the audio streams seen by the MCU.
audioRTCPSenderReports	Integer	The number of RTCP sender reports for the audio streams seen by the MCU.
audioRTCPOther	Integer	The number of other RTCP packets seen for the audio streams.
audioRTCPPacketsSent	Integer	The number of RTCP packets send by the MCU.
videoRTCPReceiverReports	Integer	As for the audio equivalents.
videoRTCPSenderReports	Integer	
videoRTCPOther	Integer	
videoRTCPPacketsSent	Integer	
currentLayout	Integer	The actual layout in use for the video stream being sent by the MCU to streaming viewers. The values for this are described in Appendix A.
layoutSource	String	One of **`family<x>`**, **`conferenceCustom`**, or **`participantCustom`**, and describes the reason for the current layout.
focusType	String	One of: **`participant`**, **`voiceActivated`** or **`h239`**.
focusParticipant	Struct	A participant identification structure (i.e. with conferenceName, participantName, participantProtocol and participantType members). Should only be present if focusType is participant.
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.

The stream structures used in the audioStreams/videoStreams responses above have the following fields:

Field	Type	Comments
codec	String	The codec in use, or **`other`** for undefined codecs.
count	Integer	The number of users of this codec.
bitRate (optional)	Integer	The bit rate of this stream in bits/second. This is only present for video streams with a defined codec.
width (optional)	Integer	The maximum width and height of this stream.

Copyright © TANDBERG 2008

height (optional)	Integer	

This method will return a fault code of "no such conference" if there is no *active* conference with the given name, regardless of the presence a configured but inactive conference of that name.

4.7 conference.streaming.modify

Modifies the current layout for streaming viewers for a conference.

Parameter	Type	Comments
conferenceName	String	Name of the conference whose layout is to be modified.
cpLayout (optional)	String	The current layout behavior for streaming viewers - see Appendix A.
borderWidth (optional)	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.
focusType (optional)	String	One of: **`voiceActivated`**, **`h239`** or **`participant`**.
focusParticipant (optional)	Struct	A participant identification structure (i.e. with conferenceName, participantName, participantProtocol and participantType members). Should only be present if focusType is participant.

4.8 conference.paneplacement.query

Queries the current pane placement configuration.

Parameter	Type	Comments
conferenceName	String	The name of the conference to be queried.

This returns a struct containing the following response fields:

Response	Type	Comments
enabled	Boolean	true if pane placement is enabled and in use; false otherwise.
panes (optional)	array of structs	This is only present if enabled above is true. The struct definition is as below.

The panes array contains structures with the following format:

Field	Type	Comments
type	String	Any one of: **`default`** - the default behaviour **`blank`** - a blank window **`loudest`** - the current loudest speaker **rolling** – shows a sequence of conference participants, changing according to the configured rolling interval **`h239`** - the h239 content channel **`participant`** - a participant as identified below.
index	Integer	The index of this pane.
participantType (optional)	String	Participant identification. Only present if this pane contains a specific participant.
participantProtocol (optional)	String	
participantName (optional)	String	

4.9 conference.paneplacement.modify

Modifies the pane placement configuration of a particular conference.

Parameter	Type	Comments
conferenceName	String	Name of the conference to be queried.
enabled (optional)	Boolean	true to enable pane placement, false to disable.
panes (optional)	array of structs	See below for format of the structures.

The panes array contains structures which define a specific pane and its contents. If a pane index is not present in the array, then that pane will remain unchanged. Participant identification is as returned in participant.enumerate.

Field	Type	Comments
index	Integer	The index of the pane to be changed.
type	String	Any one of: **default** - the default behaviour **blank** - a blank window **loudest** - the current loudest speaker **rolling** – shows a sequence of conference participants, changing according to the configured rolling interval **h239** - the h239 content channel **participant** - a participant as described in the three optional fields.
participantType (optional)	String	Participant identification. Only required if type is participant, these identify a specific participant.
participantProtocol (optional)	String	
participantName (optional)	String	

Because not all panes are guaranteed to be changed, this call returns the following structure:

Response	Type	Comments
panesModified	Integer	The number of panes successfully modified. This will be the number of elements in the panes array on complete success, and zero if there is no panes array.

4.10 participant.add

Adds a participant to a conference.

Parameter	Type	Comments
conferenceName	String	The name of the conference to which to add the participant.
participantName	String	The name of the participant to be added. This must be a unique value, i.e. not the same as any existing participant. Note that for **ad_hoc** participants, this is optional, but must be present otherwise.
participantProtocol	String	If present, must be **h323, sip** or **vnc** – these are

Parameter	Type	Comments
(optional)		the only protocols that the API can currently use.
participantType (optional)	String	If present, must be **by_address** or **ad_hoc**. Note that if this conference is an **ad_hoc** conference, this value should also be **ad_hoc**. Ad hoc participants can only be added to active conferences.
address	string (< 32 chars)	The participant's E.164 directory number, hostname or IP address.
gatewayAddress (optional)	string (< 32 chars)	IP address or hostname of an H.323 gateway.
useSIPRegistrar (optional)	Boolean	Whether to use a registrar in making this a call. (Ignored if the protocol is not sip)
transportProtocol (optional)	String	One of: **default**, **tcp**, **udp** or **tls**. (Ignored if the protocol is not sip)
password (optional)	String	The password for vnc endpoints.
deferConnection (optional)	Boolean	If true, don't call out to this participant immediately, but wait for a "participant.connect" command.
All of the following parameters are optional, and control the conferencing behaviour of the MCU with respect to the endpoint in question; for example, the maximum resolution of the video streams used, or whether the participant is able to control their conference view layout.		
maxBitRateToMCU	Integer	The maximum bit rate to the MCU specified as kBit/s.
maxBitRateFromMCU	Integer	The maximum bit rate from the MCU specified as kBit/s.
motionSharpnessTradeoff	String	One of **default** (to use the global default setting), **preferMotion**, **preferSharpness** and **balanced**.
displayNameOverrideStatus	Boolean	If true, use the specified "displayNameOverrideValue" text as the participant's display name during the conference.
displayNameOverrideValue	string (< 32 chars)	Value to use as the participant's display name (if "displayNameOverrideStatus" set to true).
cpLayout	String	This sets the initial conference view layout for the video sent to this participant. Refer to Appendix A for the full list of available layouts.
layoutControlEnabled	Boolean	Controls whether this participant is able to change the conference view layout that they see; 1 (true) means that the participant can change the layout using FECC or DTMF, 0 (false) means that they cannot.
audioRxMuted	Boolean	1 (true) means that audio from this participant will not be heard by other conference participants.
audioRxGainMode	String	One of: **none** – no extra gain applied **automatic** – automatic gain control applied **fixed** – fixed number of dBs of gain applied.
audioRxGainMillidB	Integer	If audio gain mode is **fixed**, this is the number of decibels of gain applied, multiplied by 1000, and can be a negative value.
videoRxMuted	Boolean	true means that video from this participant will not be seen by other conference participants.
videoTxWidescreen	Boolean	If true, the video sent to this participant will be in a

Parameter	Type	Comments
		form suitable for a widescreen (16:9) display.
videoTxMaxResolution	String	One of: **`cif`**, **`4cif`** or **`max`**.
videoRxMaxResolution	String	Same as above.
autoConnect	Boolean	If this is true and a participant whose e164, dns, or IP address matches this participant's address dials into the MCU, it will be moved directly to this conference. In order to stop the MCU dialing out to the participant, as the conference starts, use deferConnection.
autoDisconnect	Boolean	When set to true, the participant will be disconnected from the conference if another participant disconnects and only participants configured to be automatically disconnected remain in the conference.
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.
suppressDtmf	Boolean	Whether inband DTMF tones should be suppressed (removed) from the audio stream being received from this participant.
linkType	String	Currently, must be one of: **`cascadeSlaveToMaster`** or **`default`**. (Currently this is only taken note of if participantType is **`by_address).`**

All participants in a conference must have a "participantName" that is unique to the conference but it need not be unique across all conferences.

Participants can be added before or during a conference. A participant which is added at any time via the API will be added to the configured list of participants, and thus will be called at the start of the conference by the MCU for any conference which has any sort of repetition; to avoid this, a participant must be removed directly using participant.remove.

Note: If a "participantName" matches the name of an endpoint in the list of configured endpoints (via the web interface, go to **Endpoints**) the two are treated as unrelated. This is because in the web interface named, configured, endpoints have the "participantType" value **`by_name`**, whereas API participants are of type **`by_address`**.

4.11 participant.remove

This call removes a participant from the database of configured participants, and also removes this participant from any conferences. It will also remove all records of this participant's presence in a conference.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	

4.12 participant.modify

Depending on the operationScope parameter below, this call modifies the configuration of a participant (configuredState), or the active state of a participant in a conference (activeState).

For example, if the parameter layoutControlEnabled is included in a call to participant.modify, then the effect of the call will depend on operation scope as follows:

- If operationScope is **activeState**, the active participant's ability to control their layout will immediately change, but the configured value will remain unchanged, so that if they were to reconnect later, the state of layoutControlEnabled would revert back to how it is in the configuration.
- If operationScope is **configuredState,** the participant's current ability to control their layout will be unaffected, but their configuration will be changed so that in future occurrences of the conference (or when the participant is reconnected) they will have the newly configured state.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	
operationScope	String	One of **`activeState`** or **`configuredState`**. This parameter specifies the scope of the changes to be made, be they to the configured state of an endpoint, or to the active state of a participant in a conference.
address	String	All these parameters are optional. They should not be present if operationScope is **`activeState`**. These parameters override the configured values.
gatewayAddress	String	
useSipRegistrar	Boolean	
transportProtocol	String	
password	String	
deferConnection	Boolean	
autoConnect	Boolean	
linkType	String	
maxBitRateToMCU	Integer	
maxBitRateFromMCU	Integer	
motionSharpnessTradeoff (optional)	String	One of **`default`** (if set to the global default setting), **`preferMotion`**, **`preferSharpness`** and **`balanced`**. Optional, and valid for both **`configuredState`** and **`activeState`** scopes.
displayNameOverrideStatus	Boolean	All of these parameters are optional, and override / change the values provided in the "participant.add" call. Depending on the value of the operationScope parameter, these either, if **`configuredState`**, change the stored configuration of a participant, or, if **`activeState`**, change the active participant state, resulting in real-time changes to that participant.
displayNameOverrideValue	String	
cpLayout	String	
layoutControlEnabled	Boolean	
audioRxMuted	Boolean	
audioRxGainMode	String	
audioRxGainMillidB	Integer	
videoRxMuted	Boolean	
videoTxWidescreen	Boolean	
autoDisconnect	Boolean	
suppressDtmf	Boolean	

Parameter	Type	Comments
important (optional)	Boolean	This setting should not be present if operationScope is "configuredState". Specifies whether this participant is important.
audioTxMuted	Boolean	This setting should not be present if operationScope is "configuredState".
borderWidth (optional)	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.
focusType (optional)	String	One of: **`voiceActivated`**, **`h239`** or **`participant`**.
focusParticipant (optional)	Struct	A participant identification structure (i.e. with conferenceName, participantName, participantProtocol and participantType members). Should only be present if focusType is participant.

If there is no operationScope parameter, the MCU will attempt to change both active and configured states. This is deprecated behaviour, and should not be relied upon.

4.13 participant.connect

This method is used primarily for API-configured participants with deferConnection set to true, but can also be used to reconnect disconnected participants.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	

4.14 participant.disconnect

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	

This call causes the MCU to tear down its connection to the specified participant, if such a connection exists. This is different from "participant.remove" above because:

- in the case of configured participants, it does not remove the configuration (thus allowing later re-connection with "participant.connect")
- in the case of ad hoc participants, it does not remove the record of the previous connection

4.15 participant.move

This method is used to move a participant from one conference to another.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	

Parameter	Type	Comments
participantProtocol	String	
participantType	String	
newConferenceName	String	The name of the conference to move the participant to.

This will only move an active participant. Even if this participant is preconfigured, the configuration is unchanged.

A fault code of "no such participant" is returned if the participant is not moved or does not exist.

4.16 participant.enumerate

This method is used to return data about participants in conferences on the MCU. Several calls may be required to receive data about all participants; see the notes on enumerateID below.

Parameter	Type	Comments
enumerateFilter (optional)	String	An enumerate filter string.
enumerateID(optional)	String	The value returned by the last enumeration call. If this parameter is omitted, a new enumeration is started.
operationScope	array of strings	This should contain none, either or both of **`currentState`** or **`configuredState`**. If **`currentState`** is present, the active configuration of each participant is returned by the MCU in the currentState structure. If **`configuredState`** is present, the stored configuration is returned in the configuredState structure

Valid expressions within the enumerate filter are as follows:

Expression	Type	Comments
connected	Boolean	True if the participant is currently connected to a conference.
disconnected	Boolean	True if the participant has been connected to a conference, but is now disconnected.

Note that a participant that has not yet connected to a conference (e.g. they have deferred connection specified) is neither connected nor disconnected.

This method returns:

Response	Type	Comments
enumerateID (optional)	String	The value which should be used in the next call to get the next set of data. If this is not present, there is no further data available from the MCU.
participants	array of structs	See below for contents.

and an array called “participants” of structs which contain:

Field	Type	Comments
participantName	string	Participant identification as described above.
participantProtocol	String	

Field	Type	Comments
participantType	String	
conferenceName	String	
autoAttendantUniqueId	String	
connectionUniqueId	Integer	Corresponds to the uniqueId returned by a conference or autoattendant.
currentState (optional)	Struct	The current state of the participant as used by the MCU. Details of this struct are given below. This is only present if requested in the operationScope
configuredState (optional)	Struct	The stored configuration of the participant, if any. Details of this struct are given below. This is only present if requested in the operationScope

If the endpoint is not configured, the configuredState structure is empty; otherwise the configuredState structure contains the following entries:

Field	Type	Comments
address	String	The address used to connect to the remote endpoint in question. Only returned when the address is known.
gatewayAddress	String	
useSipRegistrar	Boolean	Whether to use a registrar in making this a call.
transportProtocol	String	One of: **`default`**, **`tcp`**, **`udp`** or **`tls`**.
password	String	The password for vnc endpoints.
deferConnection	Boolean	true if this participant's connection is being deferred.
displayNameOverrideStatus	Boolean	Indicates whether the displayName value is the result of being overridden.
maxBitRateToMCU	Integer	As for "participant.add"; in kbps.
maxBitRateFromMCU	Integer	
motionSharpnessTradeoff	String	One of **`default`** (if set to the global default setting), **`preferMotion`**, **`preferSharpness`** and **`balanced`**.
audioRxMuted	Boolean	
audioRxGainMode	String	One of **`fixed`**, **`none`** or **`automatic`**.
audioRxGainMillidB	Integer	Only returned if audioRxGainMode is **`fixed`**.
videoRxMuted	Boolean	
videoTxWidescreen	Boolean	
layoutControlEnabled	Boolean	
cpLayout	String	The configured layout behavior for this participant - see appendix A.
autoConnect	Boolean	Whether or not participants matching this address should be automatically connected to the conference.
autoDisconnect	Boolean	Whether or not the participant should be automatically disconnected from the conference when all other participants disconnect.
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.
linkType	String	One of: **`cascadeSlaveToMaster`** or **`default`**.

The currentState structure contains the following responses, if present.

Field	Type	Comments
address	String	The address used to connect to the remote endpoint in question. Only returned when the address is known (i.e. the participant was or is to be called out by the MCU, or the participant is an ad_hoc participant calling in which provided an address).
gatewayAddress	String	
ipAddress	String	The IP address to which the MCU is connected for this endpoint - this will usually be the endpoint itself, but may be a gatekeeper or gateway. Only present for active participants.
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only available after the participant has connected.
displayNameOverrideStatus	Boolean	Indicates whether the displayName value is the result of being overridden.
maxBitRateToMCU	Integer	As for "participant.add"; in kbps.
maxBitRateFromMCU	Integer	
motionSharpnessTradeoff	String	One of **`default`** (if set to the global default setting), **`preferMotion`**, **`preferSharpness`** and **`balanced`**.
callState	String	One of: **`dormant`**, **`alerting`**, **`connected`** or **`disconnected`**. When dormant, the callDirection field is not returned.
connectTime	dateTime.iso8601	Only returned after the participant is connected. This value is always present if the call state is **`connected`**. It may or may not be defined for participants in the **`disconnected`** state, depending on whether they were ever connected
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	String	Only returned after the participant has disconnected; this contains one of the disconnect reason strings given in section 8
connectPending	Boolean	true if sending a "participant.connect" command for this participant will cause either the initial connection to that endpoint (in the event that it was configured with "deferConnection" set) or a re-connection to that endpoint (in the event that it has disconnected)
audioRxCodec	String	
audioRxLost	Integer	
audioRxReceived	Integer	
audioTxCodec	String	
audioTxReportedLost	Integer	
audioTxSent	Integer	
audioRxMuted	Boolean	
audioRxGainMode	String	One of **`fixed`**, **`none`** or **`automatic`**.

Field	Type	Comments
audioRxGainMillidB	Integer	Only present if the audioRxGainMode is **`fixed`**.
audioTxMuted	Boolean	true if audio is not being transmitted to this participant.
videoRxCodec	String	
videoRxLost	Integer	
videoRxReceived	Integer	
videoTxCodec	String	
videoTxReportedLost	Integer	
videoTxSent	Integer	
videoRxMuted	Boolean	
videoTxWidescreen	Boolean	
autoDisconnect	Boolean	
important	Boolean	
activeSpeaker	Boolean	true if this participant is currently the active speaker in the conference.
layoutControlEnabled	Boolean	
activeConferenceId	String	The active conference ID of the current conference - see conference.enumerate for details of this field. This field is only present if the participant is currently in an active conference.
currentLayout	Integer	The actual layout in use for the video stream being sent by the MCU to this participant. This parameter will not be present if the participant is in an auto attendant rather than a conference, or if the MCU is not currently transmitting video to the participant in question. The values for this are those described in appendix A.
layoutSource	String	This will be one of **`family<x>`**, **`conferenceCustom`**, or **`participantCustom`**, and describes the reason for the current layout. This parameter is only present if the currentLayout parameter is also present, i.e. if the participant is in an active conference.
callDirection	String	Either **`incoming`** or **`outgoing`**. When the callState field is '**`dormant`**', the callDirection field is not returned.
previewURL	String	The location of the preview image; this is not a complete URL, and requires a prefix of http://<hostname> (where hostname is the hostname of this MCU) before it is used.
focusType	String	One of: **`participant`**, **`voiceActivated`** or **`h239`**.
focusParticipant	Struct	Only present if focusType is **`participant`**. This structure contains participant identification members (i.e. conferenceName, participantName, participantType and participantProtocol).

Field	Type	Comments
callIdentifier	base64	The h323 id of this caller.
borderWidth	Integer	0 (no border), or 1, 2, or 3 for +1/+2/+3.
autoAttendantConfiguredName	String	If this participant is connected to an auto attendant, this field holds the name of that auto attendant: the value will change as the user navigates through an MCU's configured menu structure.
mediaEncryption	String	One of the following values: **encrypted** - all media channels to and from this endpoint are encrypted **unencrypted** - all media channels to and from this endpoint are encrypted **mixed** - some channels are encrypted and some not **unknown** - none of the above; this may occur when a participant has very recently connected and no media channels have been established yet
audioRxEnergyMillidB	Integer	The measured energy of a participant's audio sent to the MCU. Typically this will be a negative value in the range -30000 (-30dB for very quiet) and 0 (very loud).
audioRxMutedRemotely	Boolean	Whether this endpoint is muted remotely.

Note: This participant information is returned for all participants added to the conference using the participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

4.17 participant.fecc

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	
direction	String	One of: `up`, `down`, `left`, `right`, `zoomIn`, `zoomOut`, `focusIn`, `focusOut`.

4.18 participant.message

Puts a message on the display of a given participant.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	
message	String	The string to send to the participant.

Parameter	Type	Comments
verticalPosition (optional)	String	Specifies where to show the message: "top", "middle" or "bottom". Message is always horizontally centered, and omitting this parameter is equivalent to choosing "middle".
durationSeconds (optional)	Integer	The length of time, in seconds, to display the message. This defaults to 30 seconds.

4.19 participant.diagnostics

Returns diagnostic information about a given participant.

Parameter	Type	Comments
conferenceName	String	Participant identification as described above.
autoAttendantUniqueId	String	
participantName	String	
participantProtocol	String	
participantType	String	

The method response returns the following:

Response	Type	Comments
videoTxFrameRate	Integer	
videoRxFrameRate	Integer	
videoRxFramesReceived	Integer	
videoTxChannelBitRate	Integer	
videoTxSelectedBitRate	Integer	
videoTxActualBitRate	Integer	
videoTxLimitReason	String	One of: **`notLimited`**, **`viewedSize`**, **`quality`**, **`aggregateBandwidth`**, **`flowControl`** or **`endpointLimitation`**.
videoRxChannelBitRate	Integer	
videoRxSelectedBitRate	Integer	
videoRxActualBitRate	Integer	
videoRxLimitReason	String	One of: **`notLimited`**, **`viewedSize`**, **`quality`**, **`aggregateBandwidth`**, **`flowControl`** or **`endpointLimitation`**.
videoTxWidth	Integer	
videoTxHeight	Integer	
videoTxInterlaced	Boolean	
videoRxWidth	Integer	
videoRxHeight	Integer	
videoRxInterlaced	Boolean	

4.20 autoAttendant.enumerate

Parameter	Type	Comments
enumerateID(optional)	String	The value returned by the last enumeration call, if omitted, a new enumeration is started.

This function has no valid enumerate filter expressions.

This method returns:

Response	Type	Comments
enumerateID(optional)	String	The value which should be used in the next call to get the next set of data. If this is omitted, then no further data is available from the MCU.
autoAttendants	array of structs	See below for contents.

and an array called "autoAttendants" of structs which contains:

Response	Type	Comments
autoAttendantUniqueID	String	
autoAttendantConfiguredName	String	If this participant is connected to an auto attendant, this field holds the name of that auto attendant: the value will change as the user navigates through an MCU's configured menu structure.
startTime	dateTime.iso8601	The time at which the auto attendant was created.

4.21 autoAttendant.destroy

Parameter	Type	Comments
autoAttendantUniqueID	String	Identifier for the auto attendant to be destroyed.

5 IP VCR methods

Methods in this section are present only where the product supports video recording and playback, such as the Codian IP VCR 2200 Series. Throughout this section, any such product is referred to as an IP VCR.

5.1 recording.callout

This replicates the call out and record functionality from the web interface.

Parameter	Type	Comments
recordingName (optional)	String	The name to be used for the recording. If no name is specified, a default name is used.
folderId (optional)	Integer	The folder in which this recording is to be placed. If not specified, the top level folder will be used.
address	String	The hostname, IP address or e164 number to connect to.
participantProtocol (optional)	String	Either **h323** or **sip**. The protocol to be used for this connection. This defaults to **h323**.
gatewayAddress (optional)	String	The address of an h323 gateway, if required. Only used if protocol is **h323**.
useSIPRegistrar (optional)	Boolean	Only valid if protocol is **sip**. Defaults to false.

This may give an operation failed fault, with a reason given in the fault string, if the operation fails. In IP VCR release 2.3 and later, if this call succeeds then the success response includes the integer "recordingId" value used to uniquely identify the new recording. This value can then be used in later recording.configure or recording.stop calls, for instance.

5.2 recording.configure

Configures a pre-existing recording. All configuration parameters are optional, although a "no changes requested" fault will occur if there are no optional parameters present.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to modify. This should be the identifier as returned by recording.enumerate.
recordingName (optional)	String	The name for the recording.
numericId (optional)	String	The numeric ID used for this recording.
pin (optional)	String	The PIN for this recording.
registerWithSIPRegistrar (optional)	Boolean	Whether to register this recording with the sip registrar.
registerWithGatekeeper (optional)	Boolean	Whether to register this recording with the h323 gatekeeper.
playbackEnabled (optional)	Boolean	Whether this recording has playback enabled.

5.3 recording.delete

This method deletes a recording from the IP VCR.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to delete. This should be the identifier as returned by recording.enumerate.

A "no such recording" fault is returned if the recording does not exist.

5.4 recording.enumerate

This function enumerates the recordings stored on the IP VCR.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.
enumerateFilter (optional)	String	A filter expression.

Valid expressions within the enumerate filter are:

Expression	Type	Comments
recordingId	Integer	The unique index of this recording
internal	Boolean	True if the recording is stored internally
external	Boolean	True if the recording is stored externally
inProgress	Boolean	True if the recording is in the process of being made

This returns the following:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.
recordings	array of structures	See below for layout of the structures.

The "recordings" array contains the following structures:

Field	Type	Comments
recordingName	String	The name of this recording.
recordingId	Integer	A unique identifier for this recording.
folderId	Integer	The unique identifier of the folder in which this recording is stored.
numericId	String	The numeric Id registered with the SIP registrar or h323 gatekeeper.
pin	String	The PIN of this recording.
status	String	The current status of the recording. This can be any of the following: **`idle`**, **`initialising`**, **`invalid`**, **`uploading`**, **`deleting`** or **`recording`**.
playbackEnabled	Boolean	True if the recording can be played back by users.
registerWithSIPRegistrar	Boolean	True if the numeric ID is registered with the SIP registrar.
registerWithGatekeeper	Boolean	True if the numeric ID is registered with the h323 gatekeeper.

5.5 recording.stop

This method stops a recording in progress on the IP VCR software version 2.2(1.15) and later. The connection between the IP VCR and the endpoint (or endpoints) involved in the call will be dropped.

Parameter	Type	Comments
recordingId	Integer	The recording ID for the recording to stop. This should be the recordingId as returned by recording.enumerate.

A "no such recording" fault is returned if the recording does not exist or is not in the process of being recorded; this method has an effect only on those recordings whose *status* value as returned by "recording.enumerate" is **recording**.

5.6 folder.enumerate

This function enumerates all subfolders of a folder.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.

This returns the following structure:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.
folders	array of structures	See below for layout of the structures.

The "folders" array contains the following structures:

Field	Type	Comments
folderName	String	The name of the folder.
folderId	Integer	A unique identifier for this folder.
parentFolderId (optional)	Integer	The unique identifier for the parent folder of this folder. This is not present if the folder has no parent (i.e. is the top level folder).
externalPath (optional)	String	The external NFS path. Only present if there is an external path configured.
exportRecordings (optional)	Boolean	Set to true if recordings in this folder are exported via NFS. Only present if there is an external path configured.
registerWithGatekeeper (optional)	Boolean	Set to true if the recordings exported externally in this folder are to be registered with the gatekeeper. Only present if there is an external path configured.
public	Boolean	Set to true if this folder is publicly accessible.
pin	String	Contains the PIN of this folder.
autoAttendantId	String	The numerical ids used to access these functions. Note that these are not the same as other instances of similar names, e.g. recordingId. These must be unique across the device.
recordingId	String	
recordingConsoleId	String	

Field	Type	Comments
pointToPointIncomingPrefix	String	The folder's configured point to point recording prefixes. These values are only present with IP VCR software version 2.3 and later.
pointToPointOutgoingPrefix	String	

6 ISDN Gateway methods

The following methods are present on ISDN gateway products, such as the ISDN GW 3220.

6.1 Common structures

Participant records

Several functions return participant records, which have the following fields:

Field	Type	Comments
uniqueId	Integer	A unique index for this participant.
protocol	String	Either **h323** (for IP) or **h320** (for ISDN).
number	String	the E164 number, IP address or DNS name of the participant.
Name	String	The name of this participant (e.g. the h323 name).
autoAttendant	boolean	True if this participant is an auto attendant running on the gateway, false otherwise.
incoming	boolean	True if the call is incoming to the gateway, false if the call is outgoing.
videoCodec	String	The video codec used for this participant.
audioCodec	String	The audio codec used for this participant.
progress	String	The state of the connection to this participant. One of: **none**, **initial**, **proceeding**, **alerting**, **connected** or **finished.** Only present for active participants.
Fecc	boolean	True if far end camera control is established, false otherwise. Only present for active participants.
ipAddress	String	The IP address if the participant. Only present for IP participants.
callIdentifier	base64	The h323 Call Identifier for this participant. Only present for h323 participants.
channelCount	Integer	The number of ISDN channels in use. Only present for ISDN participants.
channels	array of integers	The channels in use by this call. Only present for ISDN participants.

It is worth pointing out that, while in the functions below, these structures are called participantOne and participantTwo, if the call was not initiated by the web interface or API (which is not currently possible), then participantOne will be the calling party, and participantTwo the called party.

6.2 calls.active.enumerate

Returns a list of all currently active calls on the ISDN gateway.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.

This returns the following structure:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.
calls	array of struct	See below for layout of the structures.

The calls structure contains the following fields:

Field	Type	Comments
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Structures containing participant information, as define in common structures, above.
participantTwo	Struct	
startTime	dateTime.iso8601	The start time of the call.
voiceCall	Boolean	True if this is a voice-only call, false for a video call.
aggregationCall	Boolean	True if this is an aggregation call, false otherwise.
callProgress	String	The state of the call. One of: initial, calling out, connected or dying.
encryption	String	Either **all**, **some** or **none**, depending on the current encryption state of the media channels (on the IP side of the call).
ISDN encryption	String	Either **all**, **some** or **none**, depending on the current encryption state of the media channels on the ISDN side of the call.
maxDuration	Integer	The maximum duration of this call in seconds. If there is no maximum, this value is 0.
calledNumber	String	The number originally called, or **unknown** if this number is unknown.

6.3 calls.completed.enumerate

Returns completed call records from the gateway. This function takes no parameters.

Parameter	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.

This returns the following structure:

Response	Type	Comments
enumerateID (optional)	String	An enumerateID, as specified in section 2.
calls	array of struct	See below for layout of the structures.

The calls structure contains the following members:

Field	Type	Comments
uniqueId	Integer	A unique identifier for this call.
participantOne	Struct	Participant identification structures, as defined in common structures above.
participantTwo	Struct	
startTime	dateTime.iso8601	The start time of the call
endTime	dateTime.iso8601	The end time of the call.
voiceCall	Boolean	True if this is a voice-only call, false for a video call.
aggregationCall	Boolean	True if this is an aggregation call, false otherwise.
encryption	String	Either **`all`**, **`some`** or **`none`**, depending on the current encryption state of the media channels (on the IP side of the call).
ISDN encryption	String	Either **`all`**, **`some`** or **`none`**, depending on the current encryption state of the media channels on the ISDN side of the call.
maxDuration	Integer	The maximum duration of this call in seconds. If there is no maximum, this value is 0.
calledNumber	String	The number originally called, or **`unknown`** if this number is unknown.

The calling and called party are as defined above.

6.4 isdn.ports.query

Returns the current status and the settings of an ISDN port. The device.query call gives the number of ports on the ISDN gateway.

Parameter	Type	Comments
port	Integer	The port number to query. This is zero based, so if there are four ports, they are numbered 0 to 3.

This function returns the following structure:

Response	Type	Comments
port	Integer	The port number.
type	String	The interface type. One of **`e1`**, **`j1`**, **`t1`** or **`unknown`**.
mode	String	The interface mode. One of **`terminal`**, **`network`** or **`unknown`**.
layer1	Boolean	True if layer 1 is up, false otherwise.
layer2	Boolean	True if layer 2 is up, false otherwise.
enabled	Boolean	True if this port has been enabled.
bChannels	array of struct	Only present if layer2 is up. See below for definition of the structure.
lowChannel	Integer	The index of the low channel.
highChannel	Integer	The index of the high channel.
searchHighLow	Boolean	True if the search order is high to low, false if

Response	Type	Comments
		the search order is low to high.
directoryNumber	String	The directory number of this port.

The bChannels structure has the following members:

Field	Type	Comments
id	Integer	The channel index.
active	Boolean	True if this channel is active.
voice	Boolean	True if this a voice call, false if a data call. Only present if active.
incoming	Boolean	True if this call is incoming, false if outgoing. Only present if active.
calling	String	Only present if active.
called	String	Only present if active.

This function will return a "No such port" fault (24) if the port requested does not exist.

7 IP Gateway methods

The following methods are present on the IP GW 3500 Series and MSE IP GW 8350 blade.

7.1 corpdirURI.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
corpdirURI	String	The full path of the TMS address book.

7.2 corpdirURI.configure

Configures the path to the TMS address book.

Parameter	Type	Comments
corpdirURI	String	The full path of the TMS address book.

8 Deprecated messages

These messages were supported in version 1.0 of the MCU 4200 Series Management API, but have since been superseded.

8.1 *system.query*

This method is deprecated in favour of "device.query" and "conference.enumerate".

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system's current time.
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.

The method also returns a 'conferences' <array> of <struct> for each conference, where each <struct> contains the following parameters:

Parameter	Type	Comments
conferenceName	String	The name of the conference
numJoined	Integer	The number of participants that have ever joined the conference.
numLeft	Integer	The number of participants that have ever left the conference. The difference between numJoined and numLeft gives the number of current participants.

Note that until conference.destroy is called for a particular conference, the conference will remain in the list of conferences even after the conference has ended.

8.2 *conference.query*

This method is deprecated in favour of "conference.enumerate" and "participant.enumerate".

Parameter	Type	Comments
conferenceName	String	The name of the conference of interest.

The method response returns the following:

Parameter	Type	Comments
startTime	dateTime.iso8601	The time at which the conference started at or will start at.
endTime	dateTime.iso8601	The time at which the conference will end. If the conference is permanent then this parameter is absent.
pin	String	The PIN.

The method also returns a 'participants' <array> of <struct> for each conference, where each <struct> contains the following parameters:

Response	Type	Comments
participantName	String	The participant name supplied in the participant.add message.

Copyright © TANDBERG 2008

Response	Type	Comments
displayName	String	The name used by the endpoint to identify itself. This may be different to the participantName. Only returned when the participant is connected.
address	String	The address used to connect to the remote endpoint in question. Only returned when the address is known.
callState	String	One of: 'dormant' 'alerting' 'connected' 'disconnected'
connectTime	dateTime.iso8601	Only returned after the conference has begun.
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected.
disconnectReason	String	Only returned after the participant has disconnected.

Note: This participant information is returned for all participants added to the conference using the participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

8.3 conference.participant.modify

This method has been deprecated in favour of participant.modify.

See participant.modify for details of this function.

8.4 conference.participant.remove

This method has been deprecated in favour of participant.remove.

See participant.remove for details of this function.

8.5 conference.participant.add

This method has been deprecated in favour of participant.add.

See participant.add for details of this function.

8.6 participant.enumerate

While this method is not itself deprecated, there is deprecated behaviour if there is no operationScope parameter. In this case the MCU will return a participant structure with the following members:

Response	Type	Comments
participantName	String	Participant identification as described above
participantProtocol	String	
participantType	String	
conferenceName	String	
autoAttendantUniqueId	String	
address	String	The address used to connect to the remote endpoint in question. Only returned when the address is known, or if the participant is configured via the API (which requires the address to be specified when added).
gatewayAddress	String	

Response	Type	Comments
deferConnection	Boolean	
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only available after the participant has connected.
displayNameOverrideStatus	Boolean	Indicates whether the displayName value is the result of being overridden.
maxBitRateToMCU	Integer	As for "participant.add"; in kbps.
maxBitRateFromMCU	Integer	
callState	String	One of: **`dormant`**, **`alerting`**, **`connected`** or **`disconnected`**.
connectTime	dateTime.iso8601	Only returned after the participant is connected. This value is always present if the call state is **`connected`**; it may or may not be defined for participants in the **`disconnected`** call state, depending on whether they were ever connected.
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected.
disconnectReason	String	Only returned after the participant has disconnected, one of the disconnect reason values given in section 12.
connectPending	boolean	true if a "participant.connect" command is required for this participant. This parameter will cause either the initial connection to that endpoint (in the event that it was configured with deferConnection set) or a re-connection to that endpoint (in the event that it has disconnected).
audioRxCodec	String	
audioRxLost	Integer	
audioRxReceived	Integer	
audioTxCodec	String	
audioTxReportedLost	Integer	
audioTxSent	Integer	
audioRxMuted	Boolean	
audioRxGainMode	String	
audioRxGainMillidB	Integer	
videoRxCodec	String	
videoRxLost	Integer	
videoRxReceived	Integer	
videoTxCodec	String	
videoTxReportedLost	Integer	
videoTxSent	Integer	
videoRxMuted	Boolean	
videoTxWidescreen	Boolean	
important	Boolean	
activeSpeaker	Boolean	true if this participant is currently the active speaker in the conference.
layoutControlEnabled	Boolean	
cpLayout	String	The configured layout behavior for this

Response	Type	Comments
		participant - see appendix A. This parameter will be present only for participants configured via the API.
currentLayout	Integer	Actual layout in use for the video stream being sent by the MCU to this participant. This parameter will not be present if the participant is in an auto attendant rather than a conference, nor if the MCU is not currently transmitting video to the participant in question. The values for this are those described in appendix A.
callDirection	String	Either **`incoming`** or **`outgoing`**.

9 Related information sources

9.1 system.xml

While not strictly part of the XML-RPC API, some information can be retrieved from the system.xml file. This can be downloaded via HTTP as the file system.xml in the root of the unit, i.e. http://TestMCU/system.xml

This is only present on MCU, IP VCR and IP GW products.

An example is:

```
<system>
	<manufacturer>Codian</manufacturer>
	<model>MCU 4210</model>
	<serial>MRV1001SM0004B8</serial>
	<softwareVersion>1.4(1.2)</softwareVersion>
	<buildVersion>6.6(1.2)</buildVersion>
	<hostName>TestMCU</hostName>
	<totalVideoPorts>20</totalVideoPorts>
	<totalAudioOnlyPorts>20</totalAudioOnlyPorts>
	<portReservationMode>disabled</portReservationMode>
	<maxVideoResolution>cif</maxVideoResolution>
	<uptimeSeconds>2345</uptimeSeconds>
</system>
```

The meaning of the fields is:

Field	Comments
manufacturer	The manufacturer of this MCU, i.e. Codian.
model	The model of this particular MCU, e.g. MCU 4210.
serial	The serial number of this MCU.
softwareVersion	The software version currently running.
buildVersion	The build version of the software currently running.
hostName	The host name of the system.
uptimeSeconds	The number of seconds since boot.
totalVideoPorts	The total number of video ports on the MCU. (Not supported on the IP GW).
totalAudioOnlyPorts (optional)	The total number of additional audio only ports on the MCU. Only present if there are any audio-only ports. (Not supported on the IP GW).
portReservationMode	"enabled" or "disabled", depending on the Media Port Reservation configuration setting. (Not supported on the IP GW).
maxVideoResolution	The maximum video resolution for the MCU; either "cif" or "max" if larger video is enabled. (Not supported on the IP GW).

10 Required user privileges

The following table summarises which users are permitted to perform which remote management operations.

Method	Valid users
device.query	Any user with administrator privileges.
device.network.query	Any user with administrator privileges.
device.health.query	Any user with administrator privileges.
gatekeeper.query	Any user with administrator privileges.
conference.enumerate	Any user with administrator privileges.
participant.enumerate	Any user with administrator privileges.
conference.create	Any user with conference creation abilities.
conference.destroy conference.modify conference.end	The owner of the conference, or a user with "conference creation and full control" or "administrator" privilege.
conference.streaming.query	The owner of the conference, or a user with "conference creation and limited control" or higher privileges.
corpdirURI.query	Any user with administrator privileges.
corpdirURI.configure	Any user with administrator privileges.
participant.add	Any user with "administrator" privilege, the owner of the conference, or a user with "conference creation and full control" rights.
autoAttendant.enumerate autoAttendant.destroy	Any user with administrator privileges.
participant.remove participant.modify participant.move participant.diagnostics participant.fecc participant.message participant.connect participant.disconnect	If the participant is connected to an auto attendant, administrator privileges are required. If the participant is in a conference, the user must be the owner of the conference or a user with "conference creation and full control" or "administrator" privilege.
recording.callout recording.configure recording.enumerate folder.enumerate	Any user with administrator privileges.
isdn.port.query calls.active.query calls.completed.query	Any user with administrator privileges.
Deprecated methods	
system.query	Any user with administrator privileges.
conference.query	Any user with privilege "conference detail" or higher.
conference.participant.modify	As above for "participant.modify".
conference.participant.remove	As above for "participant.remove".
conference.participant.add	As above for "participant.add".

11 Fault codes

The MCU has a series of fault codes which are given when a fault occurs during the processing of an XML-RPC request. While individual method descriptions above give some indication of which faults may occur, below is a description of all possible fault codes used within this specification, and the usual interpretations.

Fault Code	Description
1	**Method not supported**. This method is not supported on this device.
2	**Duplicate conference name**. A conference name was specified, but is already in use.
3	**Duplicate participant name**. A participant name was specified, but is already in use.
4	**No such conference or auto attendant**. The conference or auto attendant identification given does not match any conference or auto attendant.
5	**No such participant**. The participant identification given does not match any participants.
6	**Too many conferences**. The device has reached the limit of the number of conferences that can be configured.
7	**Too many participants**. There are already too many participants configured and no more can be created.
8	**No conference name or auto attendant id supplied**. A conference name or auto attendant identifier was required, but was not present.
9	**No participant name supplied**. A participant name is required but was not present.
10	**No participant address supplied**. A participant address is required but was not present.
11	**Invalid start time specified**. A conference start time is not valid.
12	**Invalid end time specified**. A conference end time is not valid.
13	**Invalid PIN specified**. A PIN specified is not a valid series of digits.
14	**Unauthorised**. The requested operation is not permitted on this device.
15	**Insufficient privileges**. The specified user id and password combination is not valid for the attempted operation.
16	**Invalid enumerateID value**. An enumerate ID passed to an enumerate method invocation was invalid. Only values returned by the device should be used in enumerate methods.
17	**Port reservation failure**. This is in the case that “reservedAudioPorts” or “reservedVideoPorts” value is set too high, and the device cannot support this.
18	**Duplicate numeric ID**. A numeric ID was given, but this ID is already in use.
19	**Unsupported protocol**. A protocol was used which does not correspond to any valid protocol for this method. In particular, this is used for participant identification where an invalid protocol is specified.
20	**Unsupported participant type**. A participant type was used which does not correspond to any participant type known to the device.
21	**No such folder**. A folder identifier was present, but does not refer to a valid folder.
22	**No such recording**. A recording identifier was present, but does not refer to a valid recording.
23	**No changes requested**. This is given when a method for changing something correctly identifies an object, but no changes to that object are specified.
24	**No such port**. This is returned when an ISDN port is given as a parameter which does not exist on an ISDN gateway.
101	**Missing parameter**. This is given when a required parameter is absent. The parameter in question is given in the fault string in the format "missing parameter - *parameter_name*".
102	**Invalid parameter**. This is given when a parameter was successfully parsed, is of the

Fault Code	Description
	correct type, but falls outside the valid values; for example an integer is too high or a string value for a protocol contains an invalid protocol. The parameter in question is given in the fault string in the format "invalid parameter - *parameter_name*".
103	**Malformed parameter**. This is given when a parameter of the correct name is present, but cannot be read for some reason; for example the parameter is supposed to be an integer, but is given as a string. The parameter in question is given in the fault string in the format "malformed parameter - *parameter_name*".
201	**Operation failed**. This is a generic fault for when an operation does not succeed as required.

12 Participant disconnect reasons

These are the possible values of the “disconnectReason” field in participant information responses:

Value	Description
Unspecified	
unspecifiedError	
remoteTeardown	
localTeardown	
noAnswer	
Moved	
Rejected	
rejectedImmediately	
Busy	
Timeout	
gatekeeperError	
networkError	
protocolError	
dnsFailed	
destinationUnreachable	
gatekeeperEnded	
videoPortAllocationExceeded	
portAllocationExceeded	
disconnectAll	
incompatibleVncVersion	
failedToConnectToServer	
authenticationFailed	
serviceUnavailable	
capabilityNegotiationError	
messageQueueOverflow	
gatekeeperRequiredButAbsent	
noGatekeeperForDN	
localGatekeeperRefused	
remoteGatekeeperRefused	
remoteGatekeeperUnreachable	
remoteGatewayResources	
gatekeeperForced	
h225SocketError	
h225ProtocolError	
h225DecodeError	
h245SocketError	
h245ProtocolError	
h245DecodeError	
q931DecodeError	
q931ProtocolError	

13 References

[1]	XML-RPC, http://www.xmlrpc.com/
[2]	RFC 2616, http://www.faqs.org/rfcs/rfc2616.html

Appendix A - Conference layouts

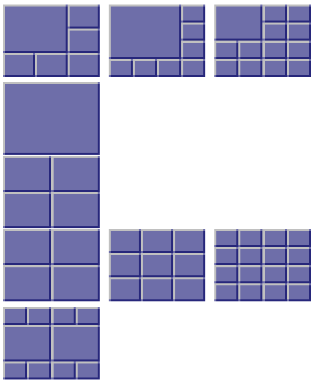
The participant.add and participant.modify methods allow a particular layout to be specified for video sent to that participant via the “cpLayout” parameter.

The “cpLayout” string parameter can take the following values:

- **default** - use the MCU’s default view family
- **family<index>** - use the specified layout family; see below
- **layout<index>** - use a specific layout; see below
- **conferenceCustom** - use the conference custom layout.

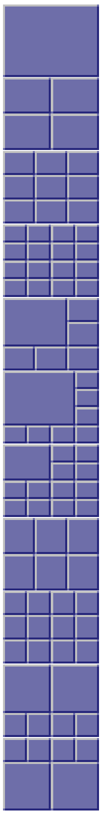
The “<index>” values for “family<index>” correspond to the pane arrangements shown below:

1
2
3
4
5

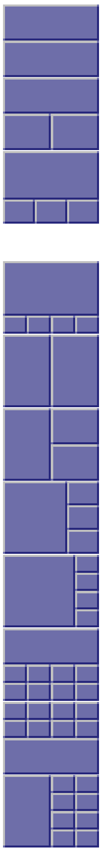


The “<index>” values for “layout<index>” correspond to the pane arrangements shown below. It is worth noting that these indices are also used for the currentLayout parameter from the participant.enumerate call.

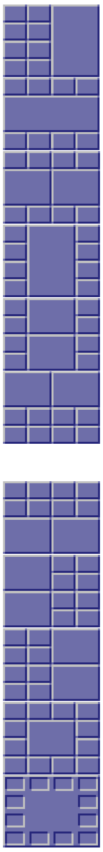
1 2 3 4 5 6 7 8 9 10 11



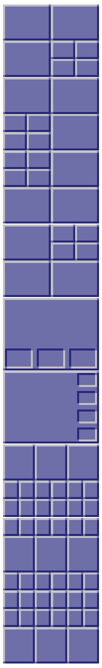
12 13 14 15 16 17 18 19 20 21 22



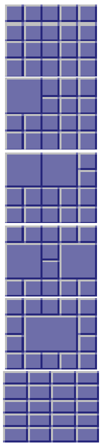
23 24 25 26 27 28 29 30 31 32 33



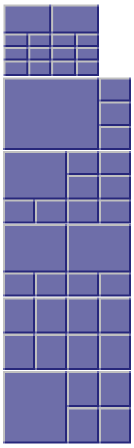
34
35
36
37
38
39
40
41
42



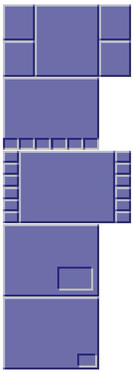
43
44
45
46
47
48



49
50
51
52
53
54



55
56
57
58
59



Appendix B - Linking conferences across MCUs

For the purposes of this description, two conferences are said to be "linked" if there is a bi-directional H.323 connection between them and each MCU is sending a video channel to the other, showing the active speaker full screen. The audio communicated between the MCUs will be the usual mix of active speakers. For clarification, the linked conferences are given different names ("linked1" and "linked2") in the explanation, but they can have the same name.

The first step is to set up the two conferences. It is important to ensure that the conferences have a numeric id set (the "conferenceID" field in "conference.create"), because, without this configured field, it is not possible to call in directly to a conference. In this example both conferences are given a numeric id, though strictly it is only necessary on the *target* MCU (i.e. the one that is called rather than the one calling).

In this specific example, "linked1" is set up on "mcu1" and "linked2" set up on "mcu2". The creation of "linked1" is shown in **example message 1**, and it is configured with numeric id "1234"; the creation of "linked2" is shown in **example message 2**, and this conference is given the numeric id "5678".

Next, a participant needs to be added to the "linked1" conference and connected to "linked2" on the target MCU. The most reliable way to accomplish this, which does not rely on the target MCU's gatekeeper usage, is to call from "mcu1" into the target conference using "mcu2" as a gateway and the target conference's numeric id as the remote address. The participant addition is shown in **example message 3** - as well as the address and gateway. It also configures the view layout to be full screen (by setting "cpLayout" to "layout1") to make sure that just the active speaker from "linked1" is sent to "linked2".

The final step is slightly more complex — it involves modifying the new "linked2" participant on "mcu2" which was the result of the call from "mcu1". The modification required is to change the view layout setting (for the video sent from "linked2" to "linked1") to full screen so that a view of the "linked2" active speaker is sent.

The complication here is that the "linked2" participant in question is not a participant created via the API, and so the API does not know the name in advance. Therefore, it is necessary to:

- poll membership of "linked2" after the connection from "linked1" has been made
- identify the participant corresponding to the call
- use its name in a "participant.modify" call to set the view layout

The simplest way to identify the participant is to look for an absence of the "address" field in a "conference.query" response: for incoming, non-API, connections this will not be present. **Example message 4** shows such a "participant.modify" call; in this case the participant name needed was "1_Codian MCU 4210".

B.1 *Example message 1 - creating conference "linked1" on "mcu1"*

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
```

```
            <string>admin</string>
          </value>
        </member>
        <member>
          <name>conferenceName</name>
          <value>
            <string>linked1</string>
          </value>
        </member>
        <member>
          <name>conferenceID</name>
          <value>
            <string>1234</string>
          </value>
        </member>
      </struct>
    </value>
  </param>
</params>
<methodCall>
```

B.2 Example message 2 - creating conference "linked2" on "mcu2"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked2</string>
            </value>
          </member>
          <member>
            <name>conferenceID</name>
            <value>
              <string>5678</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

B.3 Example message 3 - calling into "linked2" from "linked1"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>participant.add</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked1</string>
            </value>
          </member>
          <member>
            <name>participantName</name>
            <value>
              <string>remote_mcu</string>
            </value>
          </member>
          <member>
            <name>address</name>
            <value>
              <string>5678</string>
            </value>
          </member>
          <member>
            <name>gatewayAddress</name>
            <value>
              <string>10.2.1.27</string>
            </value>
          </member>
          <member>
            <name>cpLayout</name>
            <value>
              <string>layout1</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout

```
<?xml version="1.0"?>
<methodCall>
  <methodName>participant.modify</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
              <value>
                <string>linked2</string>
              </value>
          </member>
          <member>
            <name>participantName</name>
            <value>
              <string>1_Codian MCU 4210</string>
            </value>
          </member>
          <member>
            <name>operationScope</name>
            <value>
              <string>active</string>
            </value>
          </member>
          <member>
            <name>cpLayout</name>
            <value>
              <string>layout1</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

B.5 Message responses

The response to each of the above method invocations should be the same normal success indication:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value>
              <string>operation successful</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Appendix C - Revision Numbers

Note: This feature is available from API version 2.4 onwards. An application can determine the API version supported by a device from the *apiVersion* value returned in the response to a *device.query* request.

In order to reduce the size of responses when querying the MCU, the following methods support a "revision number" system:

- participant.enumerate
- conference.enumerate
- autoAttendant.enumerate

When the MCU responds to a method that supports revision numbers, it returns an extra integer field called "currentRevision". The following is an example of such a field:

```
<member>
<name>currentRevision</name>
<value>
<int>18</int>
</value>
</member>
```

The revision number is a monotonically increasing value that increases every time any query is made on the MCU via the API. In order to reduce the size of subsequent query responses, the parameter "lastRevision" may be passed in as part of a request; for example:

```
<member>
<name>lastRevision</name>
<value>
<int>18</int>
</value>
</member>
```

This indicates to the MCU that only records that have changed since this revision number should be returned. For example, in participant.enumerate, if a "lastRevision" parameter is provided, then the enumeration response will only include participants that have changed since this revision.

Note that when using revision numbers with enumerate methods, the same "lastRevision" parameter should be used for each stage of the enumeration, even though a greater "currentRevision" parameter will be returned at each stage. Not doing so could result in records which have changed not being returned. Likewise having completed an enumeration, the only "currentRevision" parameter which should be stored is the one that was returned with the first stage of the enumeration. This is the revision number that should be used as the "lastRevision" parameter next time an enumeration cycle is started.

Using the same revision numbers throughout enumerations is necessary to ensure that all records that have changed are reported, but it does mean that a record may be reported more than once occasionally when there has only been one change to it.

C.1 Discovering record removal

The problem with only returning records in responses when they have changed is that if a record is removed there is no way for the client to distinguish between it being removed and just not having changed.

There are two solutions to this problem; the first is the listAll parameter and the second is described in the next section. A client may periodically include the listAll Boolean parameter set to true to indicate that the MCU should return every record available, (enumeration limits still apply so multiple calls using the standard enumeration protocol may be required). This allows a client to resynchronize to the MCU, because it can safely assume that any record not returned by this request (or series of requests, in the case of enumerations) is no longer a record on the MCU.

For example, any participants not returned by participant.enumerate when listAll is set to true can be assumed to have been removed from the MCU.

The listAll parameter can still be used in conjunction with the lastRevision parameter: doing so means that every record will be returned, but records that have not changed since the specified revision may have many members removed from their substructures. Substructures that have had members removed in this way will contain a field named “changed” instead, which will be set to false indicating that there are no changes to the data in this substructure since the specified revision number.

C.2 Dead records

The second approach to the record removal problem is the dead parameter. The MCU will maintain a cache of records that have been removed and are in no sense considered active; a “dead” record will never be returned if revision numbers are not being used or if the listAll parameter is set to true (e.g. a previous participant record is still not considered a dead record because it would be returned by a normal participant.enumerate request).

A dead record will be returned by a method supporting revision numbers if the lastRevision parameter designates a revision at which the record was not yet dead. The returned record will contain only the fields necessary for its identification and an extra field “dead”, which will be set to true to indicate that this record should no longer considered to be present on the MCU.

These dead records are only cached on the MCU for a few minutes; therefore a client should not rely on them unless it is doing very regular polling. When using less frequent polling using the listAll parameter described above is more appropriate.

Appendix D - HTTP Keep-alives

Note: This feature is available from API version 2.4 onwards.

Another method of reducing the amount of TCP traffic when polling the MCU (see Appendix C) via the API is to use HTTP keep-alives. This method can be used with other Codian products that also support the API, such as the IP VCR and ISDN GW.

Any client which supports HTTP keep-alives may include the following line in the HTTP header of an API request:

```
Connection: Keep-Alive
```

This indicates to the Codian product that the client supports HTTP keep-alives. The device then *may* choose to not close the TCP connection after returning its response to the request. If the connection will be closed, the device returns the following line in the HTTP header of its response:

```
Connection: close
```

The absence of this line indicates that the device will keep the TCP connection open and that the client may use the same connection for a subsequent request.

The device will not allow a connection to be kept alive if:

- the current connection has already serviced a set number of requests
- the current connection has already been open for a certain amount of time
- there are already more than a certain number of connections in a "kept alive" state

These restrictions are in place to limit the resources associated with kept-alive connections. If a connection is terminated for either of the first two reasons, the client will probably find that the connection is back in a keep-alive state following the next request.

The client should never assume a connection will be kept alive.

Also note that, even after a response not containing the "connection: close" header, the connection will still be closed if no further requests are made within one minute. If requests from the client are likely to be this far apart then there is little to be gained by using HTTP keep-alives.